DETERMINING THE LOCATIONS OF THE VARIOUS CIRC RECORDING FORMAT INFORMATION BLOCKS (User data blocks, C2 & C1 words and EFM frames) ON A RECORDED COMPACT DISC

Dennis G. Howe
Optical Sciences Center
University of Arizona
Tucson, AZ 85721

January 28, 1993

Introduction:

Just prior to its being EFM modulated (i.e., converted to eight-to-fourteen channel data by the EFM encoder) and written to a Compact Disc (CD), information that passes through the CIRC Block Encoder is grouped into 33-byte blocks referred to as EFM frames. 24 of the bytes that make up a given EFM frame are user data that was input into the CIRC encoder at various (different) times, 4 of the bytes of this same EFM frame were created by the C2 ECC encoder (each at a different time), and another 4 were created by the C1 ECC encoder (again, each at a different time). The one remaining byte of the given EFM frame, which is known as the EFM frame C&D (for Control & Display) byte, carries information that identifies which portion of the current disc program track the given EFM frame belongs to and also specifies the location of the given EFM frame on the disc (in terms of a time stamp that has a resolution of 1/75th second, or 98 EFM frames). (Note: since the program track and time information is stored as a 98-byte word, a logical group consisting of 98 consecutive EFM frames must be read, and their respective C&D bytes must be catenated and decoded, before the program track identification and time position information that pertains to the entire block of 98 EFM frames can be obtained.) The C&D byte is put at the start (0th byte) of an EFM frame in real time; its placement completes the construction of the EFM frame - it is assigned just before the EFM frame enters the EFM encoder. In Fig. 1., the 33 x 1 column vector $EFM^{<i>}$ represents EFM frame No. "i"; the byte labeled as $F_{0,i}$ in this column vector is the C&D byte of EFM frame No. "i".

In the 134 (row) x 33 (column) array that accompanies this text, each row represents an EFM frame and each EFM frame's identification number is given by the value of the first element of each row, i.e., these values are found in the column identified as F_0 at the top of the 134 x 33 array. The numerical value of these (column F_0) array elements corresponds to the value of the second subscript ("i") of the element $F_{0,i}$ in the column vector $EFM^{<i>}$ of Fig. 1. EFM frames are written sequentially to the CD disc when it is recorded. Thus, byte F_0 of EFM frame No. 0 is recorded first, followed by byte D_1 of EFM frame No. 0, ..., and the 33rd byte written to the disc is byte $C1_3$ of EFM frame No. 0. As EFM frame No. 1 directly follows EFM frame No. 0, the 34th byte written to the disc is byte F_0 of EFM frame No. 1, etc. The 134 x 33 array is therefore a map of how data is written to the disc; the first row (i.e., the top row) of the array is written first, left to right, followed by the next row, again left to right, etc.

In the following discussion we will refer to four distinct blocks of data: (i) 24-byte User Input Data blocks, (ii) 28-byte C2 words, (iii) 32-byte C1 words and (iv) 33-byte EFM frames. In order to track the location on the recorded CD disc of the various data bytes, we shall "tag" (or identify) each data byte that is recorded on the disc with a number, or index, that corresponds either to the instant in time at which the particular byte entered the CIRC encoder (in the case of User Input Data), or to the instant in time at which the data byte was created by the CIRC encoder (as is the case with C2 parity bytes, C1 parity bytes and EFM frame C&D bytes). Once a byte is "tagged", the number it is tagged with will be kept, i.e., that number will not change, even when the particular byte becomes a

member of other CIRC information blocks (which will occur as the byte flows through the CIRC encoder). In order to assign numerical values to the byte "tags", we shall think of information flowing through the CIRC encoder in expanding-size blocks, namely, the User Input Data blocks, C2 and C1 words, and EFM frames mentioned above.

The User Input Data Blocks:

24 bytes of sequential user data is input to the CIRC encoder as a single 24-byte User Input Data block. A User Input Data block that enters the CIRC encoder at instant "k" is represented by the column vector $D^{<k>}$ of Fig. 1. The sequence in which user data bytes filled the User Input Data block is indicated by the first subscript of each data byte (i.e., element) of $D^{<k>}$; the instant at which the block entered the CIRC encoder is denoted by the second subscript of the various bytes of $D^{<k>}$. Thus, $D_{1,k}$ denotes the first byte of the 24-byte User Input Data block that entered the CIRC encoder at instant "k" and $D_{24,k}$ denotes the 24th (and last) byte of the User Input Data block that entered the CIRC encoder at instant "k". Sometimes we shall refer to blocks that enter the CIRC encoder at instant "k" as block No. "k", and vice versa.

The C2 Words:

Directly after they enter the CIRC encoder, the bytes of a 24-byte User Input Data block are reordered (or swapped) such that the sequence $\{D_{1,k}, D_{2,k}, D_{3,k}, ..., D_{23.k}, D_{24,k}\}$ becomes the sequence $\{D_{1,k}, D_{2,k}, D_{9,k}, D_{10,k}, D_{17,k}, D_{18,k}, D_{3,k}, D_{4,k}, D_{11,k}, D_{12,k}, D_{19,k}, D_{20,k}, D_{5,k}, D_{6,k}, D_{13,k}, D_{14,k}, D_{21,k}, D_{22,k}, D_{7,k}, D_{8,k}, D_{15,k}, D_{16,k}, D_{23,k}, D_{24,k}\}$. Then, after delaying the first 12 bytes of the reordered User Input Data block by 48 bytes (which amounts to interleaving the first half of the reordered User Input Data block by two reordered User Input Data blocks) the resultant (reordered, two-block interleaved) 24-byte block is encoded by the C2 ECC coder. This encoding process involves the computation of four C2 parity bytes and the subsequent addition of them to the reordered and doubly-interleaved 24-byte block between bytes $D_{20,k-2}$ and $D_{5,k}$. This results in a 28-byte C2 word such as the one represented by the column vector $C2^{<p>}$ in Fig. 1. Four new bytes, namely the C2 parity bytes $C2p_0$, $C2p_1$, $C2p_2$, and $C2p_3$ are created by the CIRC encoder when the C2 word is formed; we therefore "tag" these four parity bytes with a numerical value that corresponds to the instant at which these parity bytes were created. In addition, we "tag" the C2 word itself with this same numerical value. This is illustrated in Fig. 1, where it is shown that the column vector $C2^{<p>}$, which represents C2 word No. "p", contains the four parity bytes $C2p_{0,p}$, $C2p_{1,p}$, $C2p_{2,p}$ and $C2p_{3,p}$ (recall our convention is that the second byte subscript contains the numerical value that corresponds to the instant at which the C2 word is created, or equivalently, to the C2 word's No. "value"). This same C2 word (i.e., C2 word No. "p") also contains bytes from User Input Data block No. "p-2" as its first 12 information bytes (these 12 bytes experienced a delay of 2 User Input Data blocks prior to entering the C2 ECC encoder), as well as bytes from User Input Data block No. "p" as its last 12 information bytes (these 12 bytes experienced no delay between the time they entered the CIRC encoder and the time at which they entered the C2 encoder - in effect these 12 bytes were input directly into the C2 encoder).

The C1 Words:

Next, the 28-byte block output by the C2 coder is interleaved again, this time on a byte-by-byte basis. Here, the 2nd byte of a given C2 word is delayed by four C2 words (112 bytes) relative to the first byte, the 3rd byte of the given word is delayed by eight C2 words (224 bytes) relative to the first byte, ..., and the 28th byte of the given word is delayed by one-hundred-and-eight C2 words (3,024 bytes) relative to the first byte. A 28-byte block of data that has been variably-interleaved in this manner is input to the C1 ECC encoder, which computes four C1 parity bytes and adds them to the end of the variably-interleaved 28-byte block that was input to it. This is illustrated in Fig. 1, where it is shown that column vector $C1^{<n>}$, which represents C1 word No. "n", has C1 parity bytes $C1p_{0,n}$, $C1p_{1,n}$, $C1p_{2,n}$ and $C1p_{3,p}$ as its last four bytes. From Fig. 1 we also see that the 28 information bytes of C1 word No. "n" (i.e., its first 28 bytes) consist of user data bytes taken from User Input Data blocks that entered the CIRC encoder at various times ranging from two C1 word-times prior to the time of the current word (cf., the first byte of $C1^{<n>}$, namely $D_{1,n-2}$, which entered the CIRC encoder as a member of User Input Data block No. "n-2") to one-hundred-and-eight C1 word-times prior to the current word (cf., the 28th byte of $C1^{<n>}$, namely $D_{24,n-108}$, which entered the CIRC encoder as a member of User Input Data block No. "n-108"). It can also be seen from Fig. 1 that the information portion of a C1 word contains parity bytes that belong to four different C2 words (namely, bytes $C2p_{0,n-48}$ through $C2p_{3,n-60}$ of column vector $C1^{<n>}$).

The EFM frames:

A final interleaving follows the C1 encoding. The odd bytes, namely the 1st, 3rd, 5th, ..., 31st bytes, of the 32-byte block that is output by the C1 coder are each delayed by one C1 word (i.e., by 32-bytes) relative to the even bytes of the same C1 word. And lastly, a single EFM C&D byte is appended to the beginning of each of the resultant, singly-interleaved 32-byte blocks to form the 33-byte EFM frames. The column vector $EFM^{<i>}$ in Fig. 1 is a representation of EFM frame No. "i". Note that this EFM frame's C&D byte, namely byte $F_{0,i}$, and the two C1 parity bytes $C1p_{1,i}$ and $C1p_{3,i}$, are the only bytes in EFM frame No. "i" that were created (by the CIRC encoder) at instant "i". The EFM frame C&D byte $F_{0,i}$ is created by the CIRC encoder at the instant the EFM frame No. "i" is formed, and the two C1 parity bytes mentioned directly above, were created by the C1 encoder at this same time, which is why they are labeled as parity bytes that belong to C1 word No."i".

Locating Specific Information Bytes and CIRC Data Structures on the Recorded Disc:

In order to keep track of the various processing steps outlined above, we can use an index that increments once each time a new EFM frame is created. Concurrent with that event (i.e., the creation of an EFM frame), a new 24-byte User Input Data block is input to the CIRC encoding system, a new C2 word is encoded by the C2 coder, and a new C1 word

is encoded by the C1 coder. Thus, we tag all 24 bytes of a single User Input Data block with a common index, the four C2 parity bytes that correspond to a given C2 word share a common index (since they were all created at a particular time), and the four C1 parities of a given C1 word are given a common index (again, since they were simultaneously created). Specifically, the 24 bytes of a given User Input Data block are all given the index value assigned to the EFM frame C&D byte that was current when the first byte of the 24-byte User Input Data block entered the CIRC encoding system. Similarly, the group of four C2 parity bytes, and the group of four C1 parities that were each created via single encoding operations, are each given the index value of the EFM frame C&D byte that was current when those groups of parities were computed and assigned by the respective ECC coders.

It is these index values that are listed in the 134 x 33 array, which is our map of the recorded CD disc. From that array we see that the C&D byte of EFM frame No. 0 (byte F_0 in the first, i.e., top, row of the array) is assigned an index = 0, that user data byte #1 of EFM frame No. 0 (byte D_1 in row 1) came from the User Input Data block having index = -3 (i.e., this byte came from the User Input Data block that was input while the EFM frame with index = -3 was current), that user data byte #2 (byte D_2 in row 1) came from the User Input Data block having index = -6 (i.e., the User Input Data block that was input while the EFM frame with index = -6 was current), that user data byte #9 (byte D_9 in row 1) came from the User Input Data block having index = -11, ..., that C2 parity byte number 0 (byte $C2_0$ in row 1) was created by the C2 coder when the EFM frame with index = -49 was current, ..., and that C1 parity byte number 3 (byte $C1_3$ in row 1) was created by the C1 coder when the EFM frame with index = 0 was current.

The data in a given EFM frame is sequentially recorded on the disc and sequentially indexed EFM frames are recorded in sequence. Thus, the 134 x 33 array is essentially a map of how data is recorded (laid out) on the disc. Byte F_0 of EFM frame No. 0 is recorded first, followed by byte D_1 of EFM frame No. 0, etc. Byte $C1_3$ of EFM frame No. 0 is the last byte of that frame to be recorded and it is directly followed by byte F_0 of EFM frame No. 1. Because of this, the array can be used to determine the location of all the data from a specific User Input Data block, or from a specific C2 or C1 word. For example, the 24 data bytes located in columns 2 through 13 and in columns 18 through 29 that have the same index value all came from the same User Input Data block (e.g., byte D_1 in EFM frame No. 0, byte D_2 in EFM frame No. 3 and byte D_9 in EFM frame No. 8 all are members of the User Input Data block having index = -3). With the help of the column vectors given in Fig. 1, the locations on the disc of all the bytes of specific C2 and C1 ECC words can also be determined.

Using the column vector representations for an arbitrary EFM frame, an arbitrary C2 word and an arbitrary C1 word that are given in Fig. 1, we see that C2 word No. "p" (i.e., the C2 word that was created when EFM frame No. "p" was current) has data bytes that were input as part of User Input Data block No. "p-2" as its first 12 bytes and data bytes that were input as part of User Input Data block No. "p" as its last 12 bytes. Thus, if one wanted to read the disc and retrieve, for example, all the data (24 bytes) that were input as

User Input Data block No. 0, one would have to read a length of disc track which contains the entire 56 bytes that make up C2 words No. 0 and No. 2 (since the last 12 bytes of C2 word No. 0 and the first 12 bytes of C2 word No. 2 contain the 24 bytes that were input as User Input Data block No. 0). To acquire C2 word No. 0 one would have to read all those bytes in columns 2 - 13 (bytes D_1 through D_{20}) of the 134 x 33 array that have index = -2 and all those bytes in columns 14 - 29 (bytes $C2_0$ through D_{24}) that have index = 0. Similarly, to acquire C2 word No. 2, one must read all those bytes in columns 2 - 13 that have index = 0 and all those bytes in columns 14 - 29 that have index = 2. From the 134 x 33 array we see that this means that EFM frames No. 1 through No. 110 must be read. However, note that byte D_{24} of EFM frame No. 110 (which is the last byte of C2 word No. 2) is also part of C1 word No. 110, which has all of its odd-numbered bytes in EFM frame No. 111 (the column vector representation of an arbitrary C1 word given in Fig. 1 shows that the 2nd, 4th, 6th, ..., 32nd bytes of C1 word No. "n", i.e., bytes $D_{2,n-6}$, $D_{10,n-14}$, $D_{18,n-22}$, ..., $C1p_{2,n}$ are part of EFM frame No. "n", while the 1st, 3rd, 5th, ..., 31st bytes of C1 word "n", i.e., bytes $D_{1,n-2}$, $D_{9,n-10}$, $D_{17,n-18}$, ..., $C1p_{3,n}$ are part of EFM frame No. "n+1"). This means that EFM frame No. 111 must be read too, so that C1 word No. 110 can be acquired and decoded (because C1 decoding must be accomplished before C2 decoding). In total, then, 111 EFM frames must be read from the disc in order to retrieve the 24 data bytes that were input as a single User Input Data block...

To augment the understanding of how to use the 134 x 33 array to determine the locations of CIRC-encoding data structures on a recorded CD disc, on that array we have drawn circles around the locations of all 28 bytes of C2 word No. 0, and drawn squares around the locations of all 32 bytes of C1 word No. 110. Triangles have also been drawn around the locations of all 24 bytes of User Input Data block No.5. (These CIRC information blocks were arbitrarily chosen.)

CD Audio Interpolation:

The above analysis sheds light on the purpose for performing the byte reordering (swapping) that is done on the Input User Data block prior to C2 encoding. The CD system is designed, at its lowest level, as a serial (tape-like) digital audio recorder - audio program material is sampled at a rate of 44.1 KHz and those samples are digitized as 32-bit words (each 4-byte digital audio sample consists of a 16-bit right-channel audel and a 16-bit left-channel audel). Thus, a given 24-byte User Input Data block contains exactly 6 audio samples; the first four bytes (bytes $D_{1,j}$ - $D_{4,j}$, in the case of Input User frame No. "j") constitute the first audio sample, the next four bytes ($D_{5,j}$ - $D_{8,j}$) make up the second audio sample, etc. The column vector representation of a C2 word in Fig. 1 shows that the first 12 bytes of C2 word No. "p" (which consist of bytes $D_{1,p-2}$ to $D_{4,p-2}$, $D_{9,p-2}$ to $D_{12,p-2}$ and $D_{17,p-2}$ to $D_{20,p-2}$ in scrambled form) would carry the 1st, 3rd and 5th audio samples from User Input Data block No. "p-2", while the last 12 bytes of that same C2 word (which consist of bytes $D_{5,p}$ to $D_{8,p}$, $D_{13,p}$ to $D_{16,p}$ and $D_{21,p}$ to $D_{24,p}$ in scrambled form) would carry the 2nd, 4th and 6th audio samples from User Input Data block No. "p". This means that the audio samples carried in a single undecodable C2 word could be

interpolated from their "neighbor" audio samples, which are carried by two other C2 words, one of which is recorded at a distance of two C2 words earlier than the non-decodable word and the other at a distance of two C2 words later than the non-decodable word. For example, if C2 word No. 10 could not be decoded, one could interpolate values for audio samples 1, 3, and 5 that belong to User Input Data block No. 8 (and which are carried as the first 12 information bytes of C2 word No. 10) using the audio samples 2, 4, and 6 of User Input Data block No. 8 (which are carried as the last 12 information bytes of C2 word No. 8). Similarly, values for the audio samples 2, 4, and 6 from User Input Data block No. 10 (which are carried as the last 12 information bytes of C2 word No. 10) could be interpolated using the audio samples 1, 3, and 5 from User Input Data block No. 10 (which are carried as the first 12 information bytes of C2 word No. 12).

Note that the byte reordering that is done prior to C2 encoding also causes the interleaving of the left and right stereo audels that constitute the 1st, 3rd and 5th digital audio samples from a given User Input Data block. For example, from the Fig. 1 column vector representation of a C2 word, note that the first two bytes (i.e., the left channel audel) of digital audio sample 1, viz., D_1 and D_2, are followed by the first two bytes (again, the left channel audel) of digital audio sample 3, viz., D_9 and D_{10}, which are followed by the first two bytes of digital audio sample 5, viz., D_{17} and D_{18}; these are followed, in sequence, by the last two bytes (i.e., the right channel audel) of digital audio samples 1, 3 and 5 from the same User Input Data block. Digital audio samples 2, 4, and 6 from a given User Input Data block are treated similarly. This feature allows independent (separate) interpolation of values for either left channel or right channel audels using "neighbor" left channel or right channel information that is carried in another C2 word.

Handling of Long Burst Errors:

From the 134 x 33 array of recorded byte index values, we see that a burst error having minimum length of 100 bytes would be required cause two errors to occur in a single C2 word. For example, an error burst extending from byte $D_{1,3}$ (the second byte of EFM frame No. 3) to byte $D_{2,6}$ (the third byte of EFM frame No. 6) would contaminate two bytes of C2 word No. 2. We can also see that an error burst that has a minimum length of 166 bytes would be required to cause two errors in both of the two C2 words that carry "neighbor" audio samples. For example, a burst extending from byte $D_{1,3}$ to byte $D_{2,8}$ would be required to cause two errors in both C2 word No. 2 and C2 word No. 4. Clearly, very long burst errors can be corrected by the highly-interleaved C2 code, and extremely long burst errors are required to defeat the concealment of audio information errors via interpolation.

The C2 code's immunity to long burst errors, however, can readily be compromised if there is a high number of short random errors. For example, suppose that a 20-byte long burst error occurs; it will cause (only) one byte to be erroneous in each of twenty different C2 words. Then, the probability that a short random error will contaminate a second byte

of any one of these same C2 words is roughly 20 x 27 x P_B = 540 P_B, where P_B is the byte error rate that describes the occurrence of short random errors. Similarly, if a long burst error that contaminates two bytes of a given C2 word has occurred, then the probability that a short random error will contaminate a third byte of that same C2 word is roughly 26 x P_B. Clearly, if the short random error rate at the input to the C2 decoder is as high as, say $P_B = 10^{-5}$ to 10^{-3} erroneous bytes/byte (a range of rate values that may well be representative of the short random error rate of an aged, well-used CD disc), then there would be an unacceptably high probability of occurrence of uncorrectable C2 words (i.e., C2 words having more than two or three flagged errors), and/or misdecodable C2 words (i.e., C2 words having more than two unflagged errors). It is the job of the C1 code, which is decoded prior to C2 decoding, to reduce the short random error rate at the input of the C2 decoder to an acceptable level.

Correction of Short (Random) Errors:

The purpose of the single-frame interleaving that is done just prior to C1 encoding is to enhance the short random error correcting capability of the C1 code. Specifically, since the conditional probability (conditioned on the occurrence of an uncorrectable error pattern) of C1 misdecoding, i.e., having the C1 decoder unknowingly output incorrect data, is about 1% when a C1 decoder is configured to correct two errors, versus about 0.0002% when a C1 decoder is configured to correct only one error, it is desirable to use single-error correction at the C1 decoding level. (Since the Hamming distance of both the C1 and C2 codes is five, when a codeword of either of these codes is misdecoded the data block output by the decoder will have a minimum of five incorrect bytes; thus an uncorrectable error pattern that contains, say, only three erroneous bytes will be enlarged to a five, or more, byte error if erroneous decoding occurs.) The one-word interleaving done at the C1 level allows a single-error-correcting C1 decoder to correct two-byte long errors (since such errors will contaminate only one byte in each of two successive C1 words). This is important since it has been observed that the average length of the short random errors that occur in injection molded CD Digital Audio Discs and CD-ROM discs is about 11 bits in length.

The C1 block error rate (BLER) is equal to the fraction of C1 words (at the input to the C1 decoder) that contain one or more single or multiple-byte errors. If the occurrence of any type of error event is a relatively rare event, e.g., if only 0.1%, or less, of the bytes on a disc are erroneous, then the BLER provides a reasonable estimate of the rate of occurrence of error events (of any type). However, a long burst error that has length BL only increments the BLER count by at most [BL/32], where [x] represents the least integer $\geq$ x. Therefore, the BLER is essentially a measure of the rate of occurance of short random errors.

The Red Book BLER specification is BLER < 0.03. Since there are (at 1x playback speed) 7,350 C1 blocks/second, this translates to

BLER < 220 contaminated C1 words/second.

In what follows, we shall assume that an error-contaminated EFM frame experiences, on average, a single short random error event (i.e., that the short random error rate is sufficiently low that the probability of two error events occurring in a single EFM frame is very small relative to the probability of occurrence of only one error event). We shall also assume that the average length of a short random error is 1.375 bytes (11 bits). Such an average length error may start at any one of eight locations in an arbitrary byte of a depth-one interleaved C1 word. Thus, the 1.375-byte error event will contaminate only one byte in each of the two depth-one interleaved C1 words with probability 5/8 and will contaminate two bytes in one, but only one byte in the other, of the pair of interleaved C1 words with probability 3/8.

Random short error rate at the input of the C1 decoder: Since, on average, a random short error event always contaminates two C1 words, the short error event rate can be approximated by

$$\frac{BLER}{2} \ \frac{\text{short error events}}{\text{sec}} .$$

The coresponding byte error rate is

$$P_B \approx \left(\frac{BLER}{2}\right) \frac{\text{events}}{\text{sec}} \times (1.375) \ \frac{\text{erroneous bytes}}{\text{event}} \times \left(\frac{1}{7350}\right) \frac{\text{sec}}{\text{C1 word}} \times \left(\frac{1}{32}\right) \frac{\text{C1 word}}{\text{byte}}$$

or

$$P_B \approx \frac{BLER}{34{,}211} \ \frac{\text{erroneous bytes}}{\text{byte}} .$$

Taking BLER < 220 error events/sec, the random short error rate at the input of the C1 decoder is found to be

$$P_B < 6.5 \times 10^{-4} \ \frac{\text{erroneous bytes}}{\text{byte}} .$$

Random short error rate at the output of the C1 decoder: We saw earlier that, on average, a short random error event will always cause two C1 words to be contaminated and that 3/8 ths of the time one of the two contaminated words will have two bytes in error. That is, on average, only one-half of three-eighths of the contaminated C1 words will contain more than one erroneous byte. Thus, the byte error rate (due to short random errors having an average length of 11 bits) at the output of a single-error-correcting C1 decoder is given by

$$P_B \approx \left(\frac{3}{8}\right) \times \left(\frac{BLER}{2}\right) \frac{\text{undecodables}}{\text{sec}} \times \left(\frac{2}{32 \times 7350}\right) \frac{\text{erroneous bytes / undecodable}}{\text{bytes / sec}}$$

When the BLER is at its maximum specified (in the Red Book) value of 220 C1 words/second, a single-error correcting C1 decoder will fail to correct about 41 C1 words/second. And if a CD disc that is just within the Red Book BLER specification is played, the short random error rate at the output of a single-error-correcting C1 decoder is estimated to be

$$P_B \approx \left[\frac{41 \times 2}{32 \times 7350}\right] \frac{\text{erroneous bytes / sec}}{\text{bytes / sec}} = 3.5 \times 10^{-4} \frac{\text{erroneous bytes}}{\text{byte}} .$$

The short random error rate at the output of a more aggressive C1 decoder, e.g., a double-error corrector, would be substantially lower since only those C1 words that have more than two bytes in error would remain uncorrected (but, keep in mind that such a decoder will have a relatively high probability of misdecoding when an uncorrectable error pattern is encountered). Other alternatives for more powerful C1 decoding would be to use the C1 decoder to correct a single error together with one erasure per C1 input word (assuming, of course, that a means of reliably erasing bytes in the input C1 word is available), or to use the C1 decoder to correct zero (or one) errors per input C1 word and simultaneously detect up to four (two) errors and mark them as erasures for subsequent correction by an erasure-correcting C2 decoder.

The purpose of the preceding discussion of the short random error rates at both the input and output of the C1 decoder was to illustrate how the singly-interleaved C1 code contributes to the overall data reliability of the CIRC coding system, and to make the point that multiple (> 2) byte errors in C2 words (i.e., at the input to the C2 decoder) become highly probable when a long burst error occurs in conjunction with a moderately high rate of short random error events (i.e., a moderately high BLER value).

EFM frame No. "i"..

C1 word No. "n"...

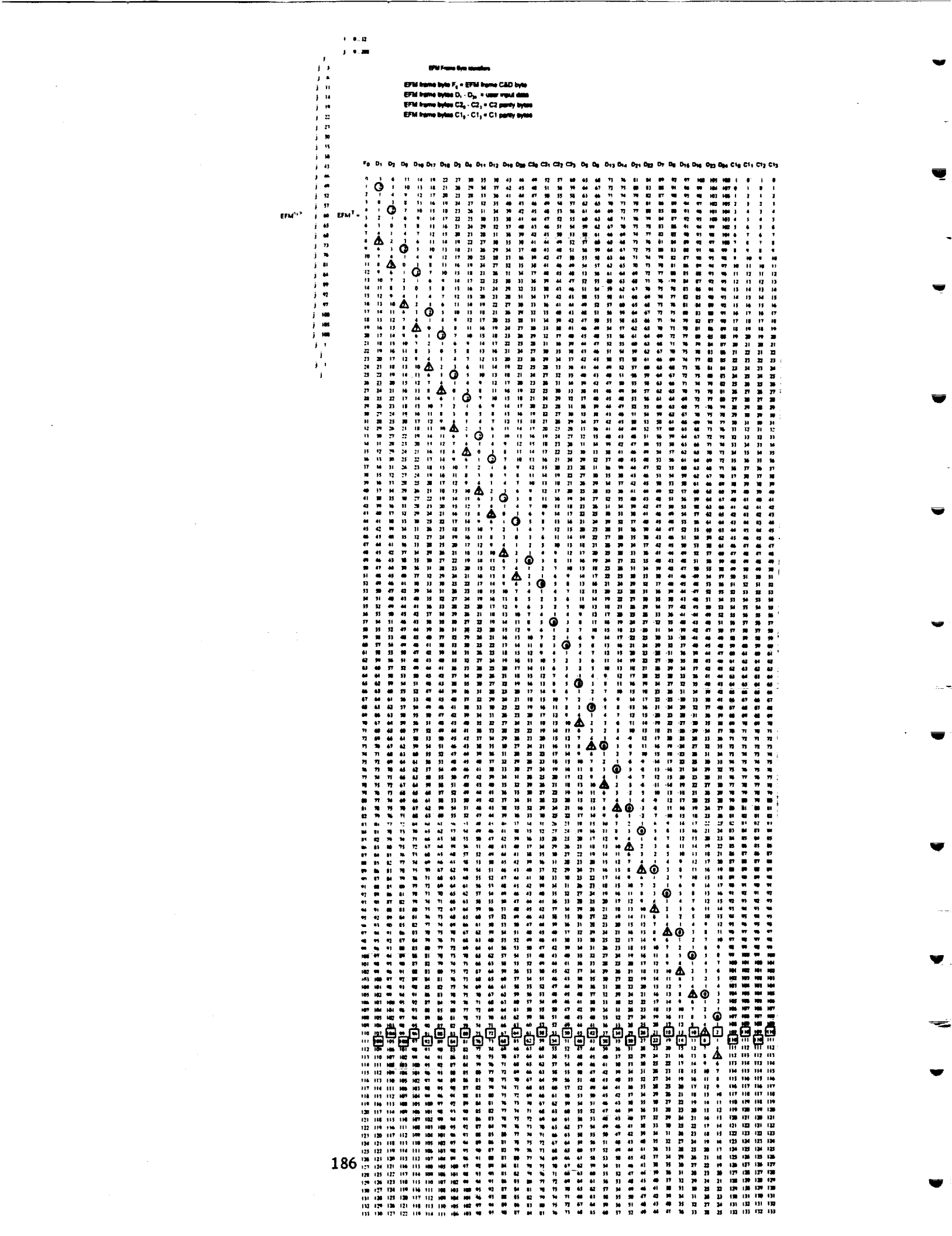
C2 word No. "p"...

User Input Data
Block No. "k"...

$$D^{<k>} := \begin{bmatrix} D_{1,k} \\ D_{2,k} \\ D_{3,k} \\ D_{4,k} \\ D_{5,k} \\ D_{6,k} \\ D_{7,k} \\ D_{8,k} \\ D_{9,k} \\ D_{10,k} \\ D_{11,k} \\ D_{12,k} \\ D_{13,k} \\ D_{14,k} \\ D_{15,k} \\ D_{16,k} \\ D_{17,k} \\ D_{18,k} \\ D_{19,k} \\ D_{20,k} \\ D_{21,k} \\ D_{22,k} \\ D_{23,k} \\ D_{24,k} \end{bmatrix} \quad C2^{<p>} := \begin{bmatrix} D_{1,p-2} \\ D_{2,p-2} \\ D_{9,p-2} \\ D_{10,p-2} \\ D_{17,p-2} \\ D_{18,p-2} \\ D_{3,p-2} \\ D_{4,p-2} \\ D_{11,p-2} \\ D_{12,p-2} \\ D_{19,p-2} \\ D_{20,p-2} \\ C2p_{0,p} \\ C2p_{1,p} \\ C2p_{2,p} \\ C2p_{3,p} \\ D_{5,p} \\ D_{6,p} \\ D_{13,p} \\ D_{14,p} \\ D_{21,p} \\ D_{22,p} \\ D_{7,p} \\ D_{8,p} \\ D_{15,p} \\ D_{16,p} \\ D_{23,p} \\ D_{24,p} \end{bmatrix}$$

$$C1^{<n>} := \begin{bmatrix} D_{1,n-2} \\ D_{2,n-6} \\ D_{9,n-10} \\ D_{10,n-14} \\ D_{17,n-18} \\ D_{18,n-22} \\ D_{3,n-26} \\ D_{4,n-30} \\ D_{11,n-34} \\ D_{12,n-38} \\ D_{19,n-42} \\ D_{20,n-46} \\ C2p_{0,n-48} \\ C2p_{1,n-52} \\ C2p_{2,n-56} \\ C2p_{3,n-60} \\ D_{5,n-64} \\ D_{6,n-68} \\ D_{13,n-72} \\ D_{14,n-76} \\ D_{21,n-80} \\ D_{22,n-84} \\ D_{7,n-88} \\ D_{8,n-92} \\ D_{15,n-96} \\ D_{16,n-100} \\ D_{23,n-104} \\ D_{24,n-108} \\ C1p_{0,n} \\ C1p_{1,n} \\ C1p_{2,n} \\ C1p_{3,n} \end{bmatrix} \quad EFM^{<i>} := \begin{bmatrix} F_{0,i} \\ D_{1,i-3} \\ D_{2,i-6} \\ D_{9,i-11} \\ D_{10,i-14} \\ D_{17,i-19} \\ D_{18,i-22} \\ D_{3,i-27} \\ D_{4,i-30} \\ D_{11,i-35} \\ D_{12,i-38} \\ D_{19,i-43} \\ D_{20,i-46} \\ C2p_{0,i-49} \\ C2p_{1,i-52} \\ C2p_{2,i-57} \\ C2p_{3,i-60} \\ D_{5,i-65} \\ D_{6,i-68} \\ D_{13,i-73} \\ D_{14,i-76} \\ D_{21,i-81} \\ D_{22,i-84} \\ D_{7,i-89} \\ D_{8,i-92} \\ D_{15,i-97} \\ D_{16,i-100} \\ D_{23,i-105} \\ D_{24,i-108} \\ C1p_{0,i-1} \\ C1p_{1,i} \\ C1p_{2,i-1} \\ C1p_{3,i} \end{bmatrix}$$

Figure 1. The CIRC encoder blocks


EFM frame byte numbers
EFM frame byte F0 = EFM frame C&D byte
EFM frame bytes D1 - D24 = user input data
EFM frame bytes C2 - C2 1 = C2 parity bytes
EFM frame bytes C1 - C1 1 = C1 parity bytes
EFM(1)
EFM(2) =